DOI: 10.5604/01.3001.0012.0703

TOOLS FOR COMPARING THE RESULTS OF THE WORK OF SORTING ALGORITHMS

Larysa Gumeniuk, Vladimir Lotysh, Pavlo Gumeniuk
Lutsk National TechnicalUniversity, Department of Automation and Computer – Integrated Technologies

Abstract. *The program implementation of sorting algorithms is obtained. The program realization of complex for comparison of sorting algorithms is obtained. Using the obtained tools, an analysis of algorithms for sorting by speed was performed depending on the number of members of the data array.*

Keywords: sorting algorithms, program realization, a software package

NARZĘDZIA DO PORÓWNANIA WYNIKÓW PRACY ALGORYTMÓW SORTOWANIA

Streszczenie. *Wykonano program realizujący algorytmy sortowania. Otrzymano programowy układ do porównania algorytmów sortowania. Wykorzystując otrzymane narzędzia, wykonano analizę algorytmów sortowania według prędkości, w zależności od liczby elementów tablicy danych.*

Słowa kluczowe: algorytmy sortowania, realizacja programu, pakiet oprogramowania

Introduction

Often, there is a need to arrange objects based on a single quality: to record number's data in ascending order, arrange people by their height, arrange words in alphabetical order. If you are able to compare any two items from the given set, then this set can always be arranged. The process of organizing information is called "sorting".

The volumes of data arrays reach the sizes that decades ago seemed almost unbelievable. The need to organize large amounts of information that is used to effectively implement a real-time search and processing procedure is increasing. The larger the amount of processed data, the more important is the task of optimizing the algorithms used, including sorting.

Thus, the development and research of methods for sorting data arrays, presenting them in a more convenient and formalized form with subsequent implementation is an urgent task at the present stage of development of high-performance computing instruments.

The purpose of this work is to develop a software package for comparing the results of the algorithms of sorting. The creation of a complex includes the development of algorithms and software for comparing the results of the algorithms of sorting.

1. Problem statement

In the development of tools (software complex) the most common algorithms for data sorting have been analysed. Due to the analysis conducted for the program implementation, the following sorting algorithms were selected:

- Built-in sorting algorithm (Python),
- Quicksort (Hoare sorting),
- Merge sort,
- Heapsort (pyramid sort),
- Binary insertion sort,
- Sorting by using simple (linear) inserts,
- Shell sort,
- Sort by choice,
- Bubble sort,
- Threaded sort,
- Bin sort (Bucket sort),
- Integer sort (Radix sort).

For program realization of selected algorithms scripting programming language is being used.

The scripting languages are used by themselves as complete base tool platforms more frequently. For example, many large commercial applications are now programmed mainly in Perl, PHP or Python. Python belongs to a dynamic typing language class, provides the programmer with an automatic "garbage collection" and convenient high-level data structures, such as dictionaries, lists, tuples, etc. Python combines striking power with a simple and understandable syntax, thought-out modularity and scalability.

The Python language interpreter is freely distributed under the Python Software Foundation (PSF) License, which is to some extent even more democratic than the GNU General Public License.

For Python there are libraries for access to the DBMS (on the Windows platform, access to the DB is possible through ADO). There are extension modules for Python under Windows and Unix/Linux for access to Oracle, Sybase, Informix, MySQL and SQLite.

The implementation of tools for comparing results is done in the Delphi programming language.

For temporarily storing data about the speed of program implementation of sorting algorithms, SQLite database is selected. The program is lined up with a library and the engine becomes an integral part of the program.

SQLite stores the entire database (including definitions, tables, indexes, and data) in a single standard file on the computer that is running the program.

Database Management System "SQLite" is a program that is provided under "open source" terms.

The SQLite library itself is written in C and is included in the Python installation application. A number of wrappers and components have been developed to work with Delphi. To implement the Delphi-SQLite connection, the ZeosLIB components have been selected.

ZeosLib is an open source project that supports multiple database management systems for Delphi, FreePascal, Kylix and BCBuilder: MySQL, PostgreSQL, Interbase, Firebird, MS SQL, Sybase, Oracle, SQLite. ZeosLib uses native DBMS libraries, but can also use its own modified libraries. Usually it's used for configuring and linking components to each other and the host.

The software implementation of the selected algorithms has been carried out in the programming language Python version 3.4.3. SQLite3 database version 3.7.0.1 was used to save data.

To save the data of calculation of the sorting algorithm time to obtain information about the average, median, and fashion, we use the box_plot database table box created using the SQLite3 DBMS.

The table structure is designed to store the data of ten runs of each sorting algorithm with a fixed value of the number of members of the Nb data array.

To analyse the obtained data a software package was developed, which includes: application for calculation (average, median, mode) and visualization of the obtained results; Application for analysis of the received data (regression equation, time) of the sorting process and their 2D and 3D visualization; an application for comparing graphs of sorting time dependence on the number of sorting elements.

Let's take a closer look at each of the applications.

2. Main results

Data Mining application – has the ability to download data from the box's database, namely the box_plot table, display it in tabular form, calculate the average, median, mode, maximum value and minimum value, and visualize this data in the form of a graph (Figure 1).

The application Chart_m is intended to calculate the total sorting time, creating the sorting time graphical dependence on the number of array elements for this sorting method (2D and 3D), maintaining the obtained dependence into the BMP file, and printing the received results (Figure 2).

The Charts application is designed to construct sorting time dependencies on the number of members of the sorting array for different sorting methods (Figure 3).

For each testing algorithm, a preliminary analysis of how much time algorithms work, depending on the size of the input data, was carried out.

It has been found by the research that all sorting algorithms, except for threaded sort, sorting by choice, sorting by simple inserts, and "bubble" sorting, work fairly quickly. For fast algorithms (built-in sort, integer sort, bucket sort, etc.) testing with incoming data up to 1,500,000 entries was performed; for others (that work slowly) – this limit was up to 110000 entries.

From the analysed data the results of the complexity of each algorithm are obtained.

Table 1 shows the dependence of the working time (sec.) on the number of elements sorted by different algorithms (complexity of the algorithm). The language of implementation is Python.

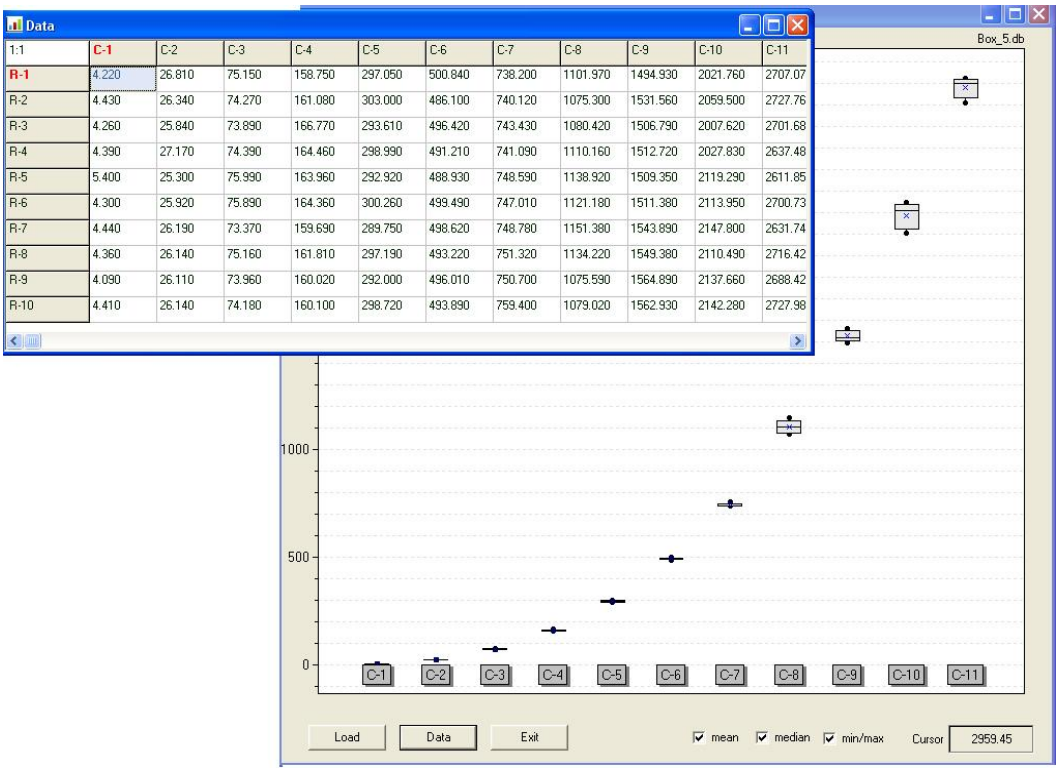

Fig. 1. Data Mining application

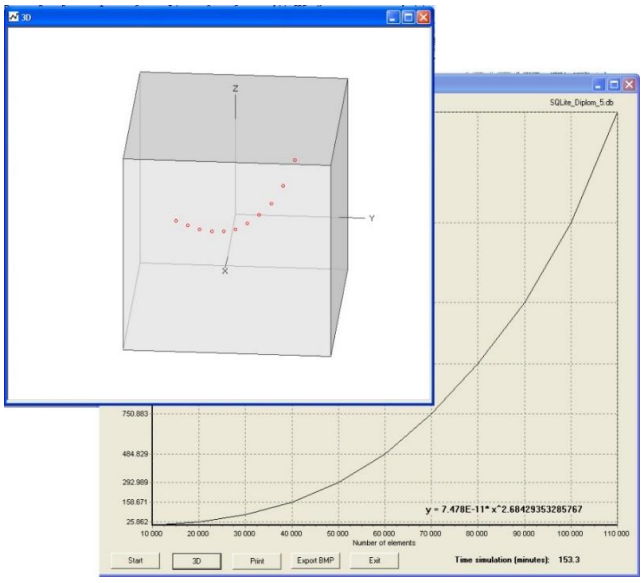

Fig. 2. The application Chart_m

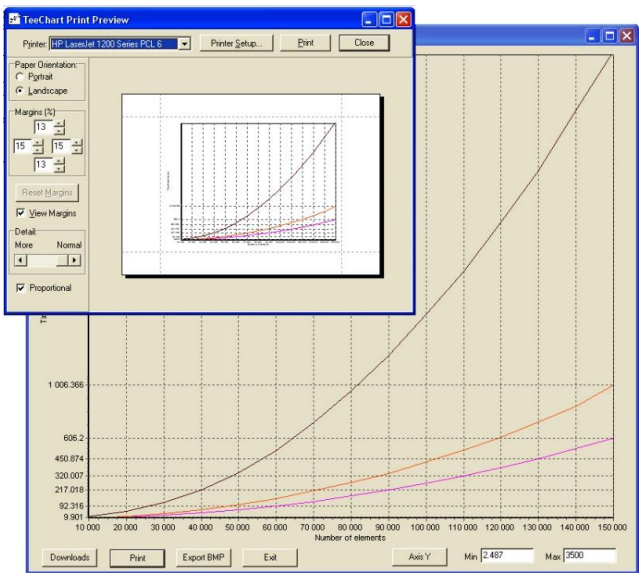

Fig. 3. The Charts application

Table 1. Dependence of the working time (sec.) on the number of elements sorted by different algorithms

Number of sorted elements	Algorithms name											
	Built-in sorting	Quicksort	Binary inserts	Merge sort	Threaded sorting	Sort by choice	Shell sort	Pyramid sort	Simple inserts	Bubble sort	Bucket sort	Sort by grade
10000	0.0083	0.092	0.277	0.1781	4.471	2.486	0.1166	0.241	13.26	4.317	0.011	0.0686
20000	0.0175	0.198	0.774	0.3781	25.862	9.901	0.2143	0.461	51.52	4.115	0.0218	0.1325
30000	0.0235	0.311	1.478	0.5867	73.461	22.26	0.3504	0.798	116.8	16.019	0.0324	0.1981
40000	0.031	0.431	2.442	0.8041	158.671	40.14	0.4584	1.077	213.3	35.415	0.0446	0.2664
50000	0.0412	0.562	3.530	1.0226	292.989	63.65	0.578	1.242	343.9	63.397	0.0571	0.3355
60000	0.055	0.691	4.828	1.2466	484.829	92.31	0.7266	1.510	512.3	101.59	0.0687	0.4042
70000	0.0567	0.817	6.258	1.4707	750.883	127.39	0.8605	1.794	722.4	146.83	0.0796	0.4759
80000	0.0661	0.959	7.937	1.7038	1087.57	169.20	0.9939	2.073	959.9	206.16	0.0924	0.5428
90000	0.076	1.087	9.834	1.9379	1502.1	217.01	1.1345	2.356	1231.7	268.33	0.1041	0.6113
100000	0.0867	1.219	11.903	2.1671	2037.53	263.05	1.2666	2.64	1539.4	339.57	0.1164	0.6812
200000	0.2051	2.614	45.241	4.6329	-	-	2.6257	5.619	-	-	0.2509	1.3716
300000	0.3316	4.067	96.237	7.1507	-	-	4.1002	8.710	-	-	0.386	2.0493
400000	0.4679	5.588	-	9.7097	-	-	5.4515	11.898	-	-	0.522	2.7254
500000	0.6084	7.111	-	12.223	-	-	7.0134	15.255	-	-	0.6609	3.4092
600000	0.7548	8.731	-	14.918	-	-	8.4102	18.398	-	-	0.7958	4.0838
700000	0.9062	10.283	-	17.796	-	-	10.236	21.748	-	-	0.9359	4.8021
800000	1.0577	11.869	-	20.516	-	-	11.721	25.116	-	-	1.0734	5.444
900000	1.2145	13.338	-	23.26	-	-	13.185	28.498	-	-	1.2114	6.1427
1000000	1.374	14.933	-	26.023	-	-	14.619	31.864	-	-	1.3499	6.8398
1500000	2.2014	23.084	-	40.095	-	-	22.002	49.392	-	-	2.0584	10.214

The following table shows that the following sorting algorithms: threaded sort, sort by choice, simple inserts, "bubble" sort, work very long in comparison with others.

The graph of the time dependence of these algorithms on the number of elements sorted is as follows (Figure 4).

From the graph it is noticeable that the algorithm of "threaded sort" is considerably inferior to others (more than 2 times).

Let's consider the running time of other algorithms.

We will start with the algorithm of binary inserts. The graph of the time dependence of this algorithm looks like this(Figure 5).

We describe the resulting curve by the equation of the form $y = ax^b$. We get y = 1.736E-8·x^1.77365. We give the similarly calculated dependencies in the four previous algorithms and in the binary insertion algorithm in Table 2.

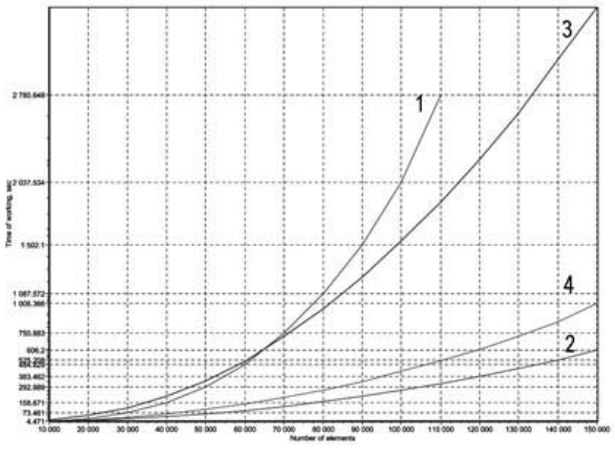

Fig. 4. Nonlinear dependence of the algorithm's running time on the number of elements that are sorted. The following algorithms are presented in the graph: 1 – Threaded sorting, 2 – Sort by choice, 3 – Simple inserts, 4 – Bubble sort

Table 2. Dependencies of the forms $y = ax^b$ for sorting algorithms

Algorithms name	Analytical equation of the curve	Coefficients	
		a	b
Threaded sorting	$y = ax^b$	7.478E-11	2,68429
Sort by choice	$y = ax^b$	1.725E-8	2,03691
Simple inserts	$y = ax^b$	5.536E-8	2.08723
Bubble sort	$y = ax^b$	3.358E-8	2,02027
Binary inserts	$y = ax^b$	1.736E-8	1.77365

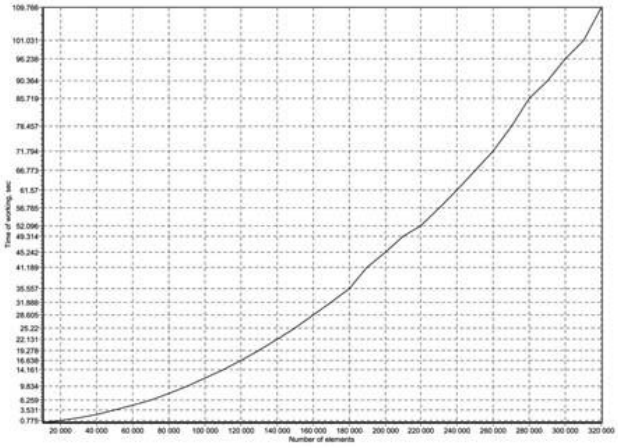

Fig. 5. The speed of the binary insertion algorithm

The binary insertion algorithm works faster than the previous four also because the constant b in this algorithm is smaller.

Let's consider a series of fast algorithms: quicksort, merge sort, Shell sort, pyramid sort (Figure 6).

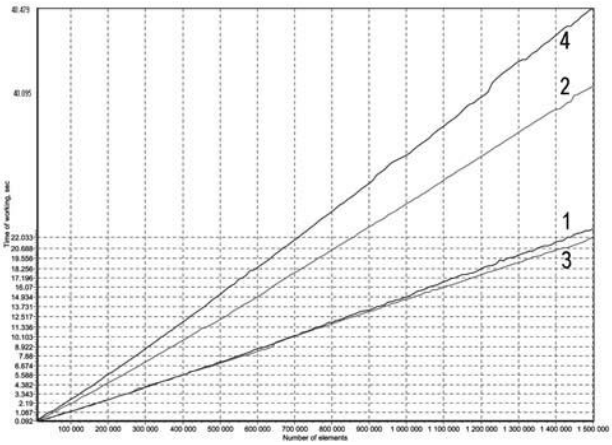

Fig. 6. The speed of algorithms: 1 – Quicksort, 2 – Merge sort, 3 – Shell sort, 4 – Pyramid sort

Accordingly, the table of coefficients of the equations for the given algorithms is as follows:

Table 3. Dependencies of the forms $y = ax^b$ for sorting algorithms

Algorithms name	Analytical equation of the curve	Coefficients	
		a	b
Quicksort	$y = ax^b$	4.151E-6	1.07562
Merge sort	$y = ax^b$	8.784E-6	1.07868
Shell sort	$y = ax^b$	6.281E-6	1.06096
Pyramid sort	$y = ax^b$	4.126E-6	1.09330

In this group the Pyramid sort Shell sort algorithm was the fastest.

The last considered algorithms:

- Sort by the built-in Python function.
- Bucket sorting.
- Sort by grade (Figure 7).

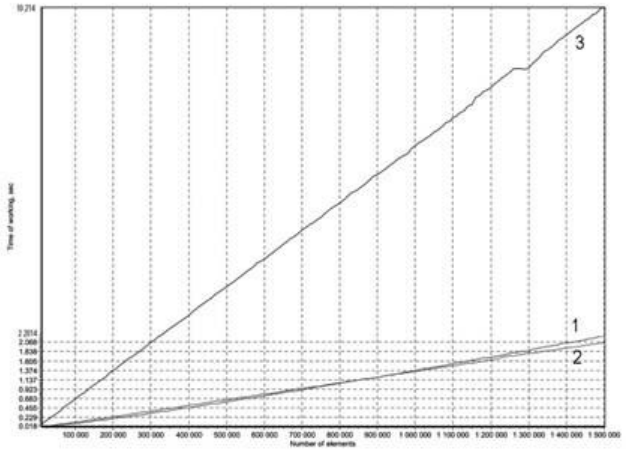

Fig. 7. The speed of algorithms: 1 – Sort by the built-in Python function, 2 – Bucket sorting, 3 – Sort by grade

Accordingly, the table of coefficients of the equations for the given algorithms is as follows:

Table 4. Dependencies of the forms $y = ax^b$ for sorting algorithms

Algorithms name	Analytical equation of the curve	Coefficients	
		a	b
Built-in Python function	$y = ax^b$	1.392E-7	1.03039
Bucket sorting	$y = ax^b$	6.609E-7	1.02246
Sort by grade	$y = ax^b$	5.478E-7	1.05756

Table 5. Total sorted table of algorithms' speed

Algorithms name	Analytical equation of the curve	Coefficients		
		a	b	b [5]
Threaded sorting	$y = ax^b$	7.478E-11	2,68429	-
Simple inserts	$y = ax^b$	5.536E-8	2.08723	2.01693
Sort by choice	$y = ax^b$	1.725E-8	2,03691	-
Bubble sort	$y = ax^b$	3.358E-8	2,02027	1.88238
Binary inserts	$y = ax^b$	1.736E-8	1.77365	2.00631
Pyramid sort	$y = ax^b$	4.126E-6	1.09330	1.08243
Merge sort	$y = ax^b$	8.784E-6	1.07868	-
Quicksort	$y = ax^b$	4.151E-6	1.07562	1.08036
Shell sort	$y = ax^b$	6.281E-6	1.06096	-
Sort by grade	$y = ax^b$	5.478E-7	1.05756	-
Built-in sorting	$y = ax^b$	1.392E-7	1.03039	1.07821
Bucket sort	$y = ax^b$	6.609E-7	1.02246	-

3. Summary

The program implementation of sorting algorithms is obtained. The program realization of complex for comparison of sorting algorithms is obtained. Using the obtained tools, an analysis of algorithms for sorting by speed was performed depending on the number of members of the data array.

References

[1] Antonova I., Karikh O.: Otsenka effektivnosti parallel'nykh algoritmov zadachi sortirovki dannykh. Promyshlennyye ASU i kontrollery 3/2010, 23–25.
[2] Kovartsev A., Popova-Kovartseva D.: Strukturnaya optimizatsiya upravlyayushchego grafa na osnove algoritma topologicheskoy sortirovki. Programmnaya inzheneriya 5/2013, 31–36.
[3] Knut D.: Iskusstvo programmirovaniya. T.3. Sortirovka i poisk. Izdatel'skiy dom "Vil'yams", Moscow 2003.
[4] Martynov V., Mironov V.: Parallel'nyye algoritmy sortirovki dannykh s ispol'zovaniyem tekhnologii MPI. Vestnik Syktyvkarskogo universiteta – Seriya 1: Matematika, Mekhanika, Informatika 16/2012, 130–135.
[5] Ovchinnikova I., Sakhnova T.: Algoritmy sortirovki pri reshenii zadach po programmirovaniyu. Informatika i obrazovaniye 2/2011, 53–56.
[6] Samun' V.: Sravneniye raboty algoritmov sortirovki, realizovannykh na yazyke Perl., 2007, http://docplayer.ru/29195102-Sravnenie-raboty-algoritmov-sortirovki-realizovannyh-na-yazyke-perl.html (available: 01.10.2017).

Ph.D. Larysa Gumeniuk
e-mail: l.gumeniuk@lntu.edu.ua

Lutsk NationalTechnical University, PhD. (technical), Head of Department of Automation and Computer – Integrated Technologies.
Research interests: Modeling ofreliability and safety ofthe automated control systems.
Has more than 60 publications in this area

Ph.D. Vladimir Lotysh
e-mail: admin@lntu.edu.ua

Lutsk National Technical University, PhD. (technical), Department of Automation and Computer – Integrated Technologies.
Scientific interests include open-source software applied for simulations of problems using distributed platforms.
Author of nearly 80 publications in this area.

Ph.D. Pavlo Gumeniuk
e-mail: p.gumeniuk@lntu.edu.ua

Lutsk NationalTechnical University, PhD. (technical), Department of Automation and Computer – Integrated Technologies.
Research interests: programming, robotics.

otrzymano/received: 21.10.2017 *przyjęto do druku/accepted: 11.05.2018*